



Trabalhando com a Canvas API

Dessa vez vamos trabalhar com desenhos bitmap e muito JavaScript!

Na unidade passada conversamos sobre representações vetoriais e imagens bitmap. Vimos que podemos usar os elementos em SVG para gerar imagens vetoriais. Nessa unidade veremos como gerar imagens em bitmap interativas diretamente no navegador por intermédio da API Canvas. Inclusive, usaremos muitos códigos JavaScript para isso!

Dentro do HTML5 o elemento canvas irá abranger, em forma de pixels em bitmap, todos os objetos contidos dentro dele. Com esse recurso, podemos criar composições gráficas 2D como linhas, formas, preenchimentos, texto e etc, todas processadas como bitmap

Essa API é indicada para projetos com composição gráfica e efeitos visuais sofisticados, como nos jogos web. Nesse caso, sua exibição é processada no navegador por intermédio de um elemento de marcação <canva>.

Os navegadores utilizam recursos de aceleração gráfica por GPU (*Graphics Process Unit*) para renderizar os projetos. Isso acelera o processamento gráfico, diminuindo o processo da CPU do usuário. Ou seja, o processamento das imagens é feito diretamente no dispositivo cliente; a Canvas API se estende a dispositivos móveis e qualquer agente que possa processar o HTML5.

Veremos nessa unidade como trabalhar com essa API, métodos e propriedades do JavaScript para criar e manipular objetos e representações gráficas em 2D.



Utilização básica do Canvas

A Canvas API fornece recursos para desenhos e gráficos de alto desempenho interpretados no dispositivo cliente com base em blocos de código JavaScript. Dessa forma, precisamos acessar o elemento <canvas> no DOM.

Dentro de nosso documento HTML precisamos inserir o elemento <canvas> propriamente dito. Por exemplo, dentro de <body> </body> podemos inserir:

```
<canvas id="game-canvas" width="200" height="200"></canvas>
```

Nesse exemplo, vamos inserir simplesmente uma superfície de desenho branca e quadrada de 200 por 200 pixels em nossa página web.

Além do atributo id, o elemento <canvas> tem apenas dois atributos *width* e *height*. Se não forem especificados, o canvas terá 300 pixels de largura por 150 pixels de altura. O elemento também pode ser redimensionado por CSS. Não esqueça de sempre fechar o elemento com o **</canvas>**.

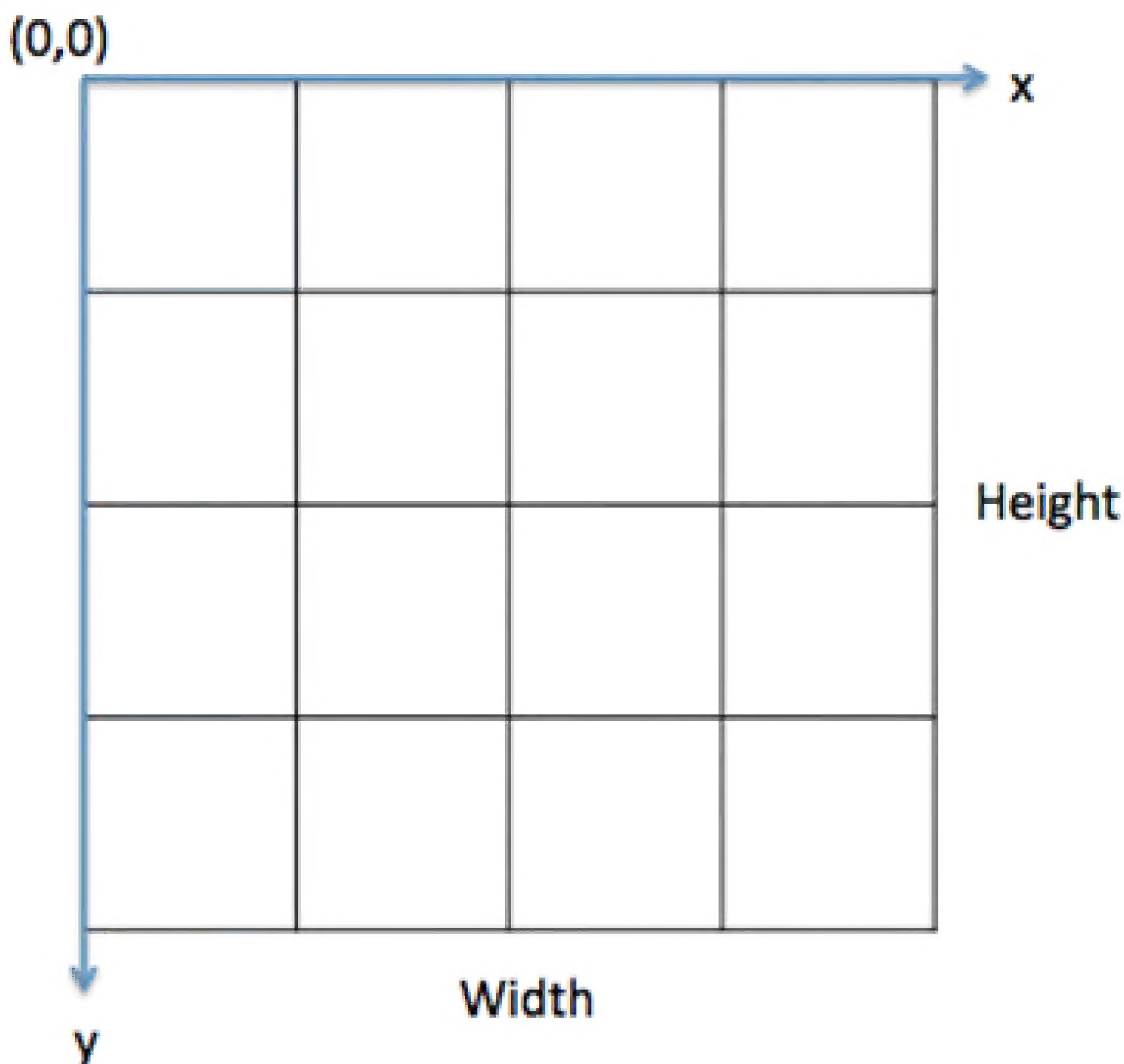
Tendo o nosso elemento <canvas> em seu devido lugar, precisamos utilizar o JavaScript para acessar e obter o contexto 2D que nos permitirá criar representações gráficas. Então para acessar o exemplo anterior, faríamos o seguinte treco de código em JavaScript:

```
// Acessar o elemento canvas no documento HTML
let canvas = document.getElementById("game-canvas");
// Obter o context 2d e ter acesso à API para desenhar
let ctx = canvas.getContext("2d");
```

Com isso, agora temos o context 2d da API Canvas armazenada na variável ctx. Isso nos dá acesso aos métodos completos da API para começarmos a desenhar e até mesmo criar animações poderosas!

Desenhando retângulos

Para que possamos fazer os desenhos, devemos estar atentos à grade de tela ou espaço de coordenadas, assim como tivemos no SVG. Na imagem a seguir temos uma representação desta grade.



Normalmente 1 unidade na grade corresponde a um pixel na tela. A origem desta grade está posicionada no canto superior esquerdo, nas coordenadas (0,0). Todos os objetos que formos desenhar estarão dispostos em relação a esta origem.

Então para desenhar formas em retângulo na API Canvas, temos os seguintes métodos para desenhá-los:

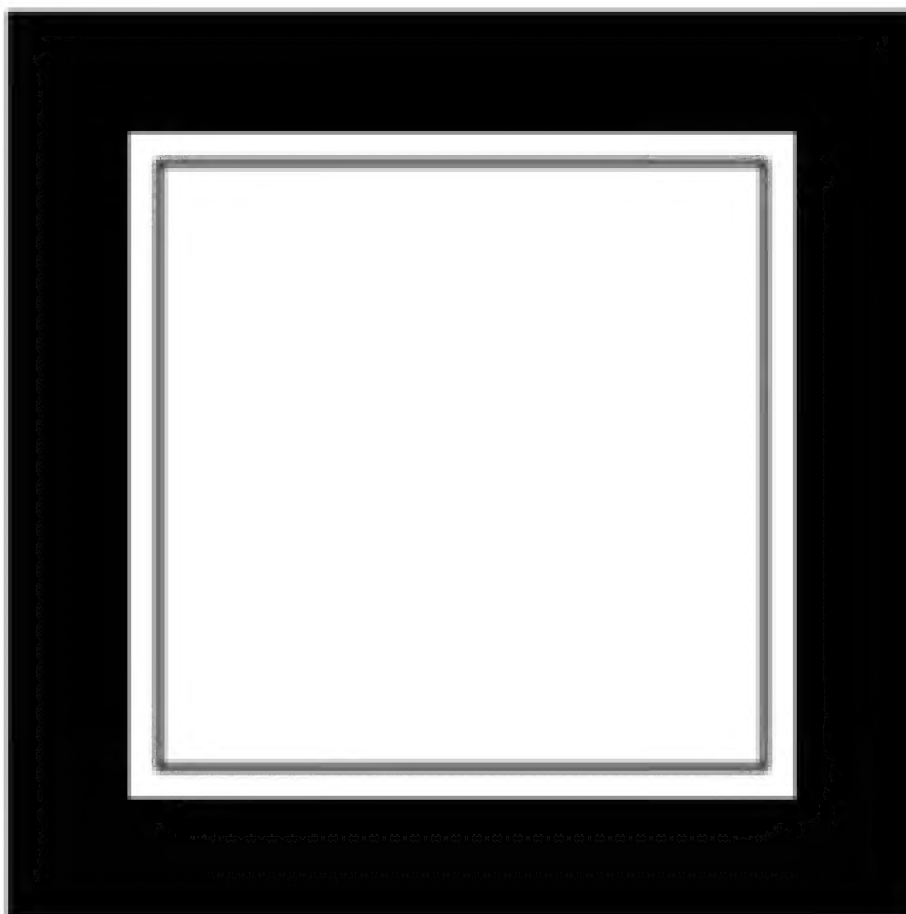
Método	Descrição
fillRect(x, y, width, height)	Desenha um retângulo preenchido.
strokeRect(x, y, width, height)	Desenha a borda do retângulo.
clearRect(x, y, width, height)	Limpa uma área de retângulo específica, tornando-a transparente.

Cada um dos métodos recebem os mesmos parâmetros: x e y determinam a posição em relação a origem, no canto superior esquerdo do retângulo. O tamanho do retângulo é então dado por *width* (largura) e o *height* (altura).

Um exemplo utilizando esses métodos em conjunto pode ser o seguinte bloco de código:

```
// Acessar o elemento canvas no documento HTML
let canvas = document.getElementById("game-canvas");
// Obter o contexto 2d e ter acesso completo à API
let ctx = canvas.getContext("2d");
// Usar os métodos para representar retângulos
ctx.fillRect(25, 25, 150, 150);
ctx.clearRect(45, 45, 110, 110);
ctx.strokeRect(50, 50, 100, 100);
```

No exemplo, como é possível notar, utilizamos os três métodos vistos para desenhar representações gráficas de retângulos. O resultado desse código são 3 formas diferentes de retângulos sobrepostos, como na imagem a seguir.



Para desenhar os objetos com cores, devemos alterar a propriedade *fillStyle*, informando a cor que desejamos utilizar. A cor pode ser nas cores padrões do HTML ou em hexadecimal. A sintaxe é simples, veja um exemplo prático para desenhar um retângulo vermelho:



```
// Acessar o elemento canvas no documento HTML
let canvas = document.getElementById(“game-canvas”);
// Obter o contexto 2d e ter acesso completo à API
let ctx = canvas.getContext(“2d”);
// Definir uma cor ao desenho, poderia em valor hexadecimal
ctx.fillStyle = "red";
// Desenhar um retângulo em
ctx.fillRect(25, 25, 150, 150);
```

Inserindo textos

Para inserirmos um texto em nossa área de desenho definido pelo <canvas> usaremos o método *fillText()*, em conjunto com as propriedades as propriedades *fillStyle* e *font*. Esse método do contexto 2d da API Canvas tem a seguinte sintaxe:

fillText(txt, x, y);

Onde txt será a mensagem a ser apresentada no navegador; x e y determinam a posição em relação a origem, no canto inferior esquerdo do objeto texto.

Como exemplo, vamos modificar o exemplo anterior inserindo um texto longo sobre o retângulo. Faremos isso com a fonte Arial com tamanho de 20 pixels e cor azul escura:

```
// Definir a cor como azul escuro
ctx.fillStyle = "#145DA0";
// Definir a fonte
ctx.font = "20px Arial";
// Escrever o texto
ctx.fillText("Olá pessoal!", 25, 20);
```

Inserindo imagens



Podemos inserir imagens com a API Canvas. Podemos usar isso para compor interfaces em nossos jogos ou para criarmos *sprites* animados. É possível utilizar os mesmos formatos suportados pelo navegador, como PNG, GIF ou JPEG.

Neste resumo, usaremos a seguinte sintaxe:

drawImage(img, x, y, width, height);

Onde img especifica a imagem a ser apresentada; x e y determinam a posição em relação a origem, no canto superior esquerdo do objeto texto; e *width* (largura) e *height* (altura) são o tamanho da imagem em pixel.

Importar imagens no canvas é um processo de duas etapas: antes de usar o método *drawimage()*, precisamos criar dinamicamente imagem HTML, juntamente com o seu caminho (endereço URL). Vejamos o exemplo a seguir:

```
// Acessar o elemento canvas no documento HTML
let canvas = document.getElementById("game-canvas");
// Obter o contexto 2d e ter acesso completo à API
let ctx = canvas.getContext("2d");
// Criar o objeto de imagem HTML
let img = new Image();
// Primeira etapa: Criar a referência para o url da imagem
img.src = "sprite-npc.jpg";
// Segunda etapa: registrar o manipulador de eventos para quando houver
// o carregamento da imagem, ela seja inserida no contexto do canvas
img.addEventListener ("load", function() {
   ctx.drawImage(img, 30, 30, 110, 110);
});
```

Explicando o exemplo: precisamos criar um objeto de imagem antes de tudo, para isso usamos o *new Image()*. Esse objeto de imagem precisa seu caminho definido e, para isso, alteramos a sua propriedade src. Quando alteramos essa propriedade, o navegador dispara um evento do tipo *onload* da imagem. Sendo assim, registramos um manipulador, fazendo com que o programa execute o bloco de código que irá inserir a imagem dentro do contexto 2D do canvas com *drawImage()*.

Agora estamos munidos de mais conhecimentos para criar interfaces programáticas com JavaScript, além do conhecimento visto nesta unidade ser essencial para trabalharmos com *sprites*, *tilesets* e animações 2D.

Vamos em frente!

Atividade Extra

Para fixar o conteúdo sobre o canvas, acesse na biblioteca virtual o livro *Canvas HTML5: composição gráfica e interatividade na web* do autor Roque Fernando marcos Souza e leia o capítulo 3 – Desenhando sobre o CANVAS, praticando os exemplos apresentados pelo autor.

Referência Bibliográfica

MDN Web Docs. **CANVAS tutorial.** Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/Canvas_API/Tutorial>.

SOUZA, R. F. M. **Canvas HTML5: composição gráfica e interatividade na web**. Brasport: Rio de Janeiro. 2013.



Ir para exercício